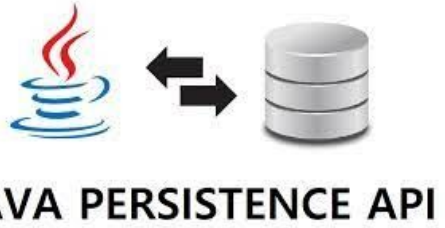


Introducción a JPA

UNJu - Programación Orientada a Objetos

Ing. José Zapana

Java Persistence API

- API estándar de Java para persistencia de objetos.
- Define un conjunto de interfaces y anotaciones.
- No es una implementación ➜ necesita un proveedor (Hibernate, EclipseLink, OpenJPA).
- Objetivo: simplificar el acceso a datos y el mapeo objeto-relacional.

Conceptos clave de JPA

- **Entidad:** clase Java que se mapea a una tabla de la base de datos.
- **Campo persistente:** atributo de la clase que corresponde a una columna.
- **EntityManager:** interfaz para realizar operaciones CRUD.
- **Unidad de persistencia (persistence.xml):** define la configuración (DB, proveedor, etc.).
- **JPQL (Java Persistence Query Language):** lenguaje de consultas orientado a objetos.

Anotaciones básicas

- **@Entity** ➜ Marca la clase como entidad persistente.
- **@Id** ➜ Indica la clave primaria.
- **@GeneratedValue** ➜ Estrategia de generación de IDs.
- **@Column** ➜ Configura el nombre o características de la columna.
- **@Table** ➜ (Opcional) nombre de la tabla.
- **@Temporal** (TemporalType.DATE / TemporalType.TIMESTAMP)
- **@Transient** ➜ Marca a un atributo para que no se persista.
-

Ejemplo

```java
@Entity
@Table(name = "usuarios")
public class Usuario {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nombre;
    private String email;
}
```

Ciclo de Vida de las Entidades

- **Nuevo (transient):** objeto creado en memoria pero no está registrada en el EntityManager ni existe en la base de datos.
- **Gestionado (managed):** objeto asociado a un contexto de persistencia. JPA la monitorea automáticamente con la base de datos al finalizar la transacción.

```
em.getTransaction().begin();
em.persist(p);
em.getTransaction().commit();
```

- **Separado (detached):** entidad que ya no está asociada al contexto.
- **Eliminado (removed):** marcado para borrarse en la BD.

Operaciones con EntityManager

- persist(obj) ➜ Inserta.
- find(Entidad.class, id) ➜ Busca por id.
- merge(obj) ➜ Actualiza.
- remove(obj) ➜ Elimina.
- createQuery("JPQL...") ➜ Consultas personalizadas.

JPA con Spring Boot y Java 21

- JPA sigue vigente y es parte de Jakarta EE (ahora bajo el nombre Jakarta Persistence).
- La versión estable más utilizada es Jakarta Persistence 3.1 (incluida en Spring Boot 3.x).
- La configuración de persistence.xml se realiza en **application.properties / application.yml**
- Spring Boot trae Spring Data JPA encima de JPA ➜ simplifica aún más (CrudRepository, JpaRepository).

Gestión de una Entity

- EntityManager: existe, pero en Spring Boot se utilizan repositorios.
- Solo es necesario agregar los método específicos de esta entidad

```
@Repository
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {
     List<Usuario> findByNombre(String nombre);
     Optional<Usuario> findByNombreIgnoreCase(String nombre);

}
```

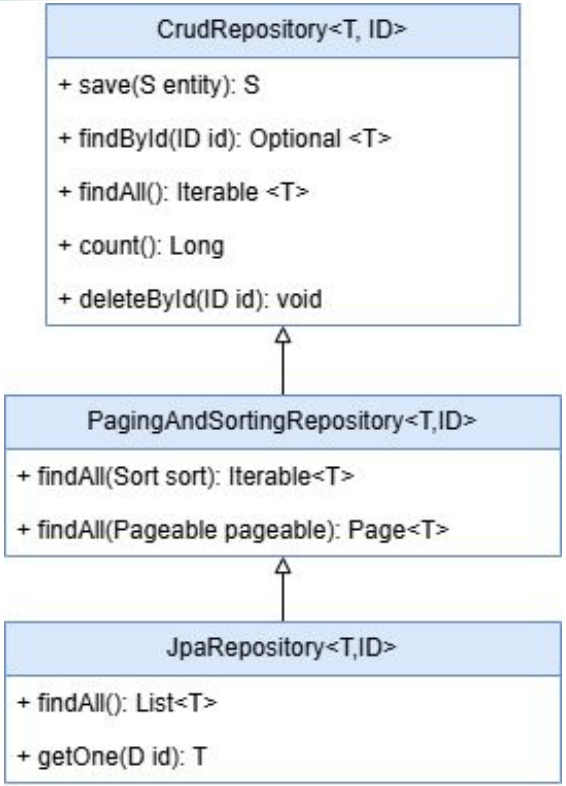
```
//Ejemplo con optional
Optional<Usuario> userOptional = usuarioRepository.findByNombreIgnoreCase("Pepe");

// Manejo seguro
Usuario user = userOptional.orElseThrow(() -> new RuntimeException("Usuario no encontrado"));
```

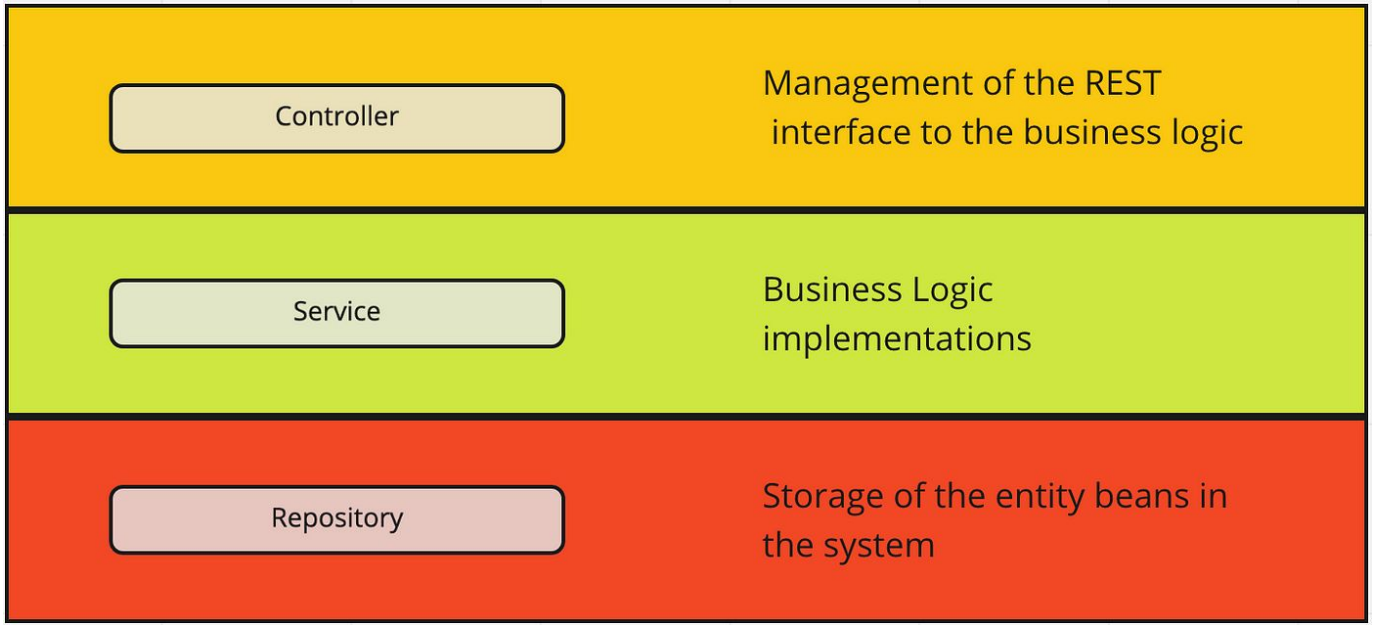
JpaRepository

Inyecta las siguientes operaciones

- count(): long
- delete(id: ID): void
- delete(entity: T): void
- delete(entities: Iterable<?extends T>): void
- deleteAll(): void
- findOne(id: ID): T
- findAll(ids: iterable<ID>): Iterable<T>
- findAll(): Iterable<T>
- save(entity: T): T
- save(entities: iterable<T>): Itereable<T>



Capas de una aplicación Spring Boot



Crear una aplicación con spring boot

- https://start.spring.io/

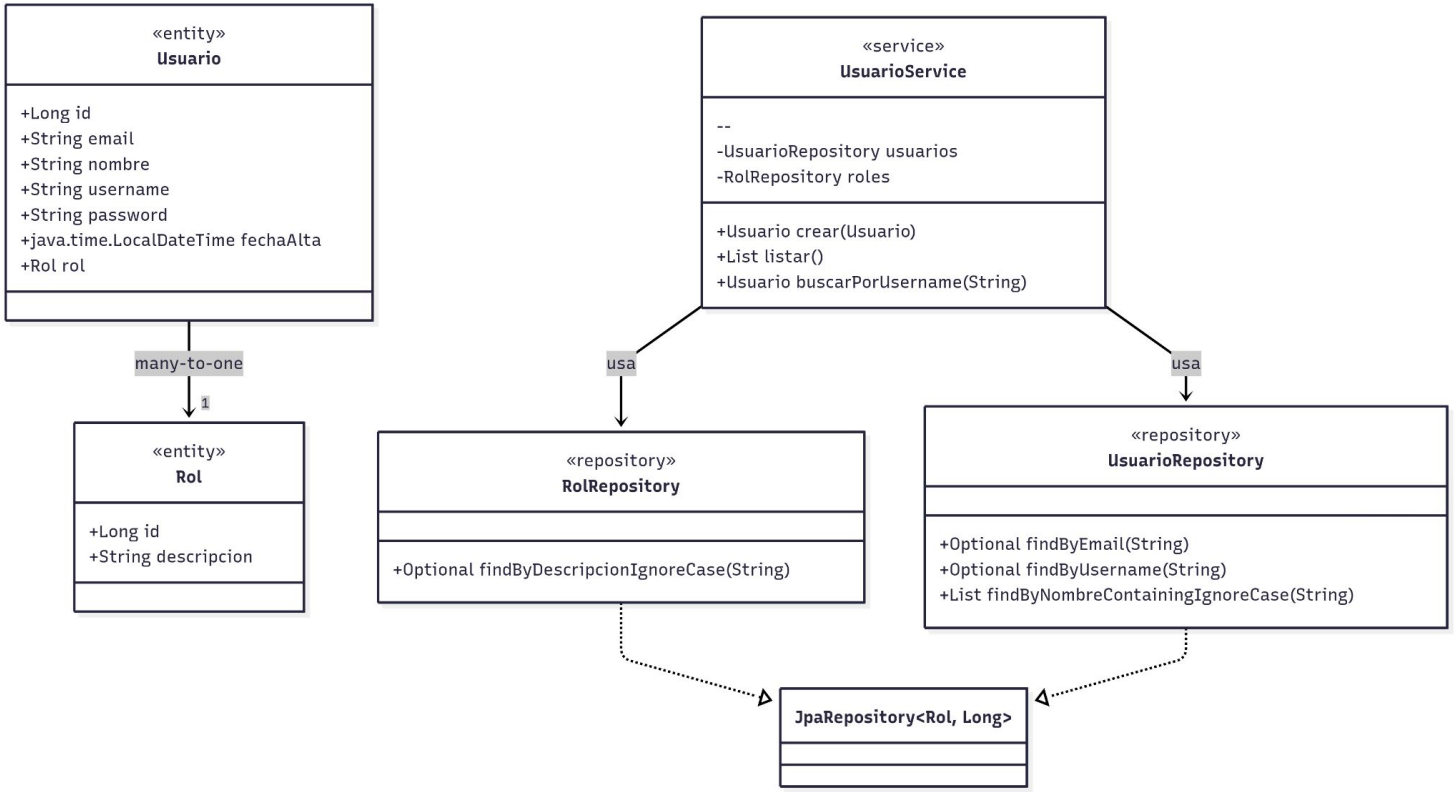
Dependencia	¿Para qué sirve?
Spring Web	Para crear controladores REST y manejar peticiones HTTP
Spring Data JPA	Para trabajar con entidades, repositorios y persistencia con JPA/ Hibernate
Lombok	Para reducir código repetitivo (getters, setters, constructores, etc.)
H2 Database	Base de datos en memoria ideal para pru
MariaDB Driver	Conector JDBC para usar MariaDB como

Dependencia	¿Por qué incluirla?
Spring Boot DevTools	Recarga automática y herramientas útiles para desarrollo
Validation (Bean Validation)	Para usar anotaciones como `@NotNull`, `@Size`, etc. en tus entidades
Spring Boot Actuator	Para monitorear métricas y salud de la aplicación (útil en producción)

Servicios

- Contienen la lógica de negocio de la aplicación
- Son el nexo entre los controladores y los repositorios DAO
- Sus métodos utilizan a los DAOs para realizar las operaciones con la Base de Datos
- Analice el ejemplo citado en el link de la última diapositiva

Ejemplo JPA - Overview



Conexión con la base de datos

application.properties (ejemplo)

```
spring.jpa.hibernate.ddl-auto=create
spring.datasource.url=jdbc:mysql://localhost:3306/champions?serverTimeZone=GMT-3
spring.datasource.username=root
spring.datasource.password=admin
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.show-sql=true
```

Referencias

- **Spring Boot — Documentación oficial:** https://docs.spring.io/spring-boot/index.html
- **Spring Data JPA — Referencia oficial:** https://docs.spring.io/spring-data/jpa/reference/index.html
- **Jakarta Persistence (JPA) API — Documentación oficial:** https://jakarta.ee/specifications/persistence/2.2/apidocs/
- **Java Persistence API (Oracle) — Descripción de la API:** https://www.oracle.com/java/technologies/persistence-jsp.html
- **MySQL Connector/J — Guía para desarrolladores:** https://dev.mysql.com/doc/connector-j/en/
- **MySQL Connector/J — Descarga oficial:** https://dev.mysql.com/downloads/connector/j/